Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №1**

по дисциплине: «Компьютерные системы моделирования»

Тема: «Построение аналитической модели по текстовому описанию задачи и по результатам экспериментов»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-22.06  Иванов А.М. |
| Проверила:  старший преподаватель кафедры ПО  Корнеева Е.И. |

Тверь 2025

**Оглавление**

[**Теория по варианту** 3](#_Toc188729465)

[**Задача 1** 4](#_Toc188729466)

[**Постановка задачи** 4](#_Toc188729467)

[**Данные** 4](#_Toc188729468)

[**Построение математической модели** 4](#_Toc188729469)

[**Решение задачи** 5](#_Toc188729470)

[**Основные результаты** 8](#_Toc188729471)

[**Вывод** 8](#_Toc188729472)

[**Задача 2** 9](#_Toc188729473)

[**Постановка задачи** 9](#_Toc188729474)

[**Данные** 10](#_Toc188729475)

[**Основные результаты** 10](#_Toc188729476)

[**Выводы** 15](#_Toc188729477)

[**Общий вывод по лабораторной работе** 17](#_Toc188729478)

[**Ссылка на реализацию (github):** 20](#_Toc188729479)

# **Теория по варианту**

Основное требование к модели — её **адекватность**. То есть соответствие модели оригиналу, результаты моделирования подтверждаются и могут служить основой для прогнозирования процессов.  
Оно также включает:

**Точность** – степень точности копирования свойств объекта моделью.

**Цель**– для решения какой задачи создается модель.

**Управляемость**– например, число параметров модели.

**Целостность** – степень слаженности подсистеме модели.

**Робастость** – устойчивость модель и по отношению к исходным данным.

**Продуктивность** – полезность модели, способность проверить адекватность модели на практике (например, если погрешность измерения характеристики объекта намного меньше точности модели, то модель непродуктивна)

А также сложность, наглядность и универсальность

# **Задача 1**

## **Постановка задачи**

На швейной фабрике для изготовления четырех видов изделий используется ткань трех артикулов. Для каждого вида изделия известны:

* нормы расхода ткани (м²),
* общее количество ткани каждого артикула на складе (м²),
* стоимость одного изделия данного вида (руб.).

Цель задачи: определить, сколько изделий каждого вида необходимо произвести, чтобы стоимость изготовленной продукции была максимальной.

## **Данные**



## **Построение математической модели**

**Целевая функция:**  
Максимизация общей стоимости продуфкции:

F(x) = 9x\_1 + 6x\_2 + 4x\_3 + 7x\_4

**Ограничения:**

1. Ограничение по ткани I:

x\_1 + 2x\_3 + x\_4 ≤ 180

1. Ограничение по ткани II:

x\_2 + 3x\_3 + 2x\_4 ≤ 210

1. Ограничение по ткани III:

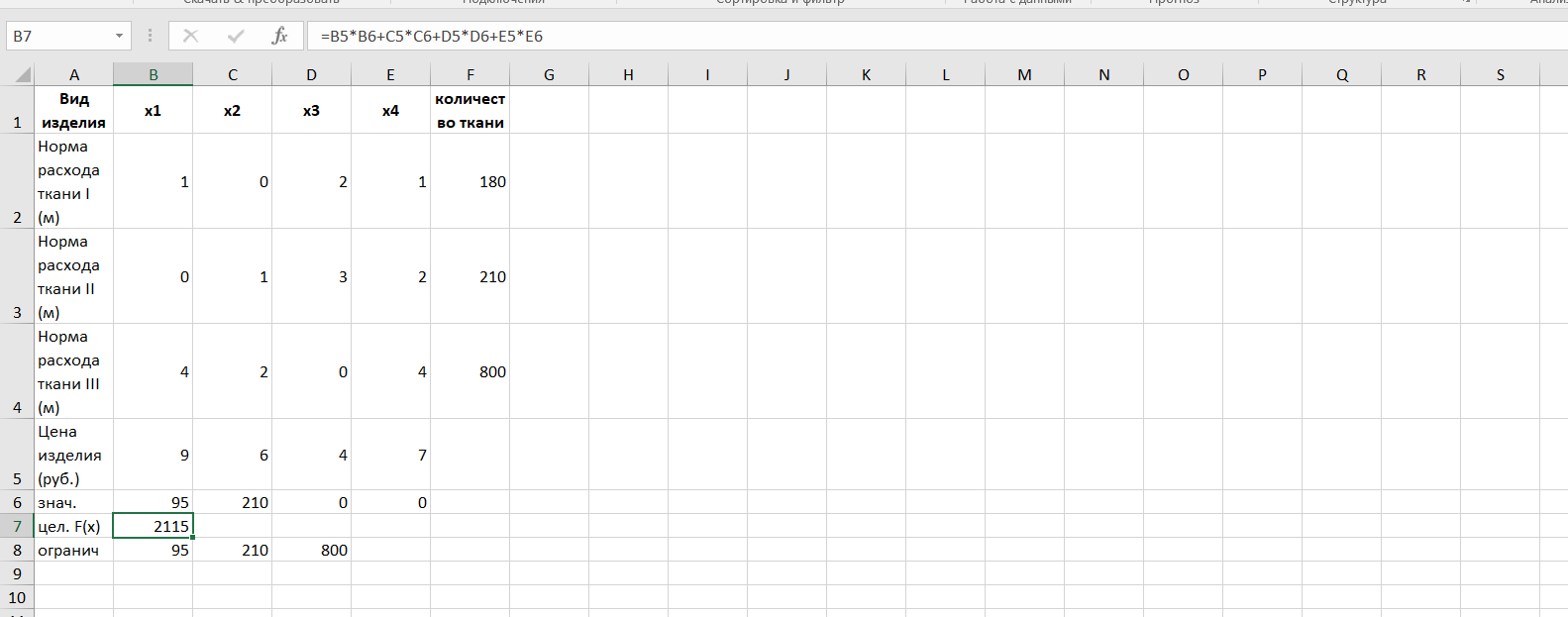
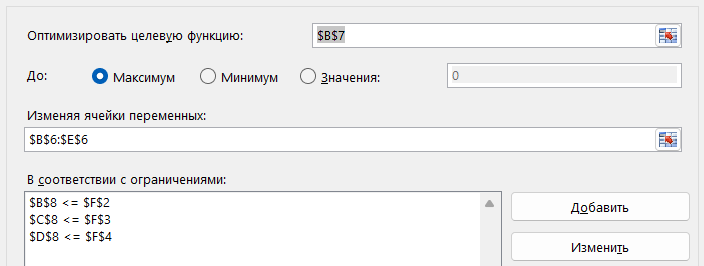
4x\_1 + 2x\_2 + 4x\_4 ≤ 800

**Переменные:**  
x\_1, x\_2, x\_3, x\_4 ≥ 0, целочисленные.

## **Решение задачи**

**Решение в MS Excel**

**Модель задачи в Excel:**

**Результат решения:**



Т.е. максимальная прибыль составит 2115 руб. при выпуске 95 изделий типа 1, и 210 изделий типа 2

**Решение с помощью Python**

Для решения задачи использовалась библиотека pulp.

Код:

from pulp import LpMaximize, LpProblem, LpVariable

model = LpProblem(name="maximize-production-cost", sense=LpMaximize)

x1 = LpVariable(name="x1", lowBound=0, cat="Integer")

x2 = LpVariable(name="x2", lowBound=0, cat="Integer")

x3 = LpVariable(name="x3", lowBound=0, cat="Integer")

x4 = LpVariable(name="x4", lowBound=0, cat="Integer")

model += 9 \* x1 + 6 \* x2 + 4 \* x3 + 7 \* x4, "Total cost"

model += x1 + 2 \* x3 + x4 <= 180, "Constraint for fabric I"

model += x2 + 3 \* x3 + 2 \* x4 <= 210, "Constraint for fabric II"

model += 4 \* x1 + 2 \* x2 + 4 \* x4 <= 800, "Constraint for fabric III"

status = model.solve()

print(f"Status: {model.status}")

print(f"Optimal production:")

print(f"x1 (item 1): {x1.value()}")

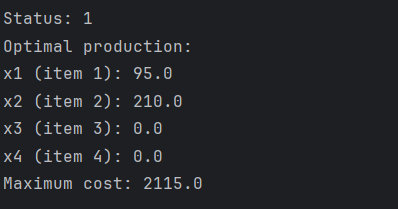
print(f"x2 (item 2): {x2.value()}")

print(f"x3 (item 3): {x3.value()}")

print(f"x4 (item 4): {x4.value()}")

print(f"Maximum cost: {model.objective.value()}")

**Результат:**



* x\_1 = 95 (изделия вида 1),
* x\_2 = 210 (изделия вида 2),
* x\_3 = 0 (изделия вида 3),
* x\_4 = 0 (изделия вида 4),
* Максимальная стоимость продукции: F(x) = 2115 руб.

## **Основные результаты**

1. **Оптимальная стратегия производства:**
   * Произвести 95 изделий вида 1,
   * Произвести 210 изделий вида 2,
   * Изделия видов 3 и 4 не производить.
2. **Максимальная стоимость продукции:** 2115 руб.

## **Вывод**

Поставленная задача успешно решена двумя способами: с использованием Excel и Python. Результаты совпадают, что подтверждает корректность модели. Использование Python обеспечивает автоматизацию и масштабируемость для решения более сложных задач.

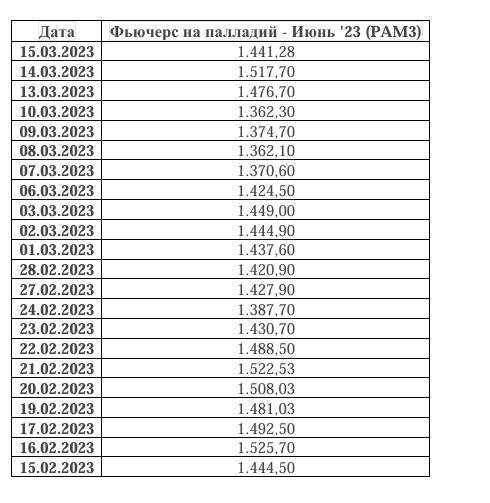
**Задача 2**

## **Постановка задачи**

По данным о котировках с сайта **Investing.com** за 15.03.2023 необходимо:

1. Построить аналитическую модель на основе данных.
2. Создать таблицу в MS Excel или Google Sheets и построить линии тренда: линейную, полиномиальную, степенную и логарифмическую. Вывести на графике уравнение и коэффициент детерминации R^2.
3. Построить аналогичные графики с использованием Python и библиотек matplotlib/seaborn. Сравнить результаты и сделать выводы.

## **Данные**



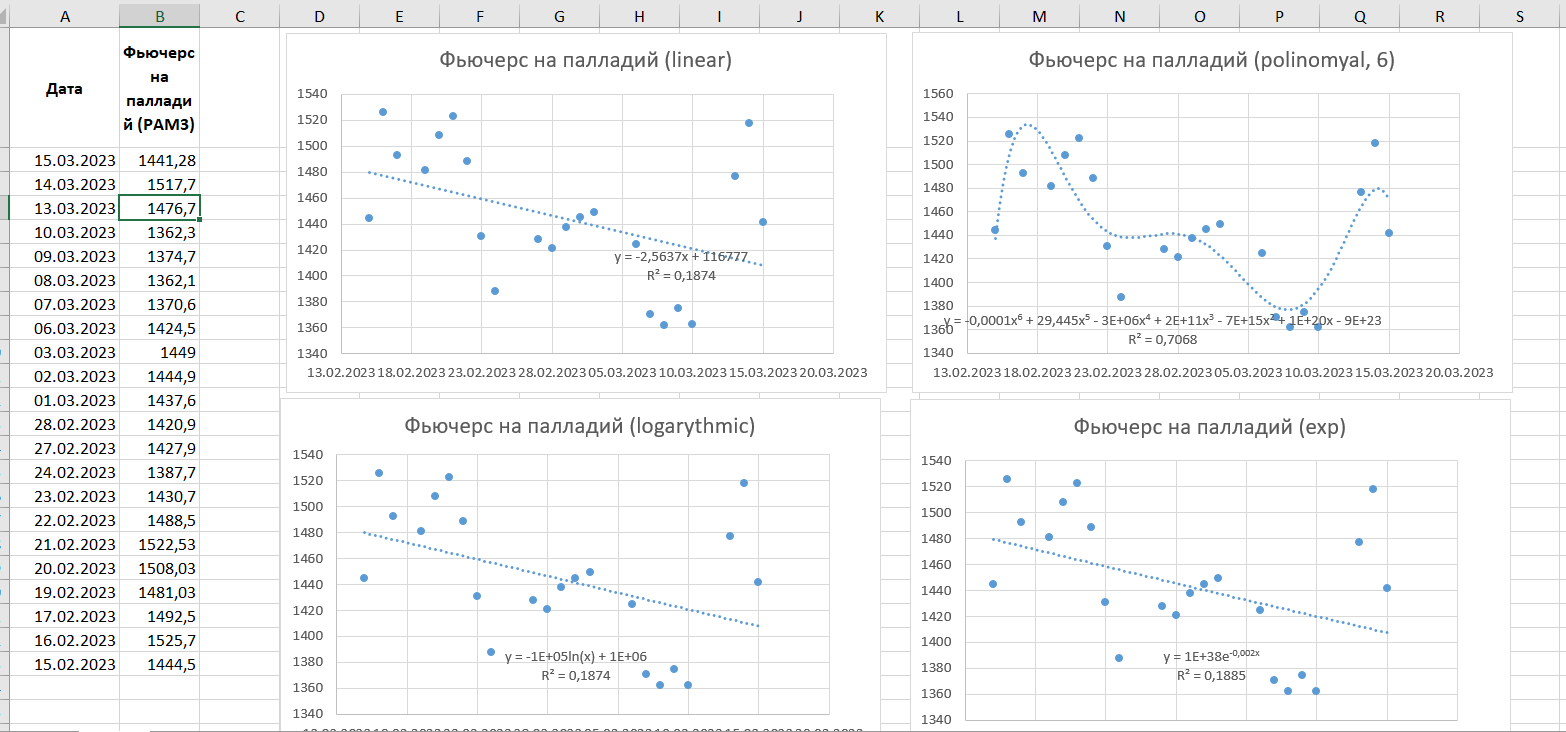
## **Основные результаты**

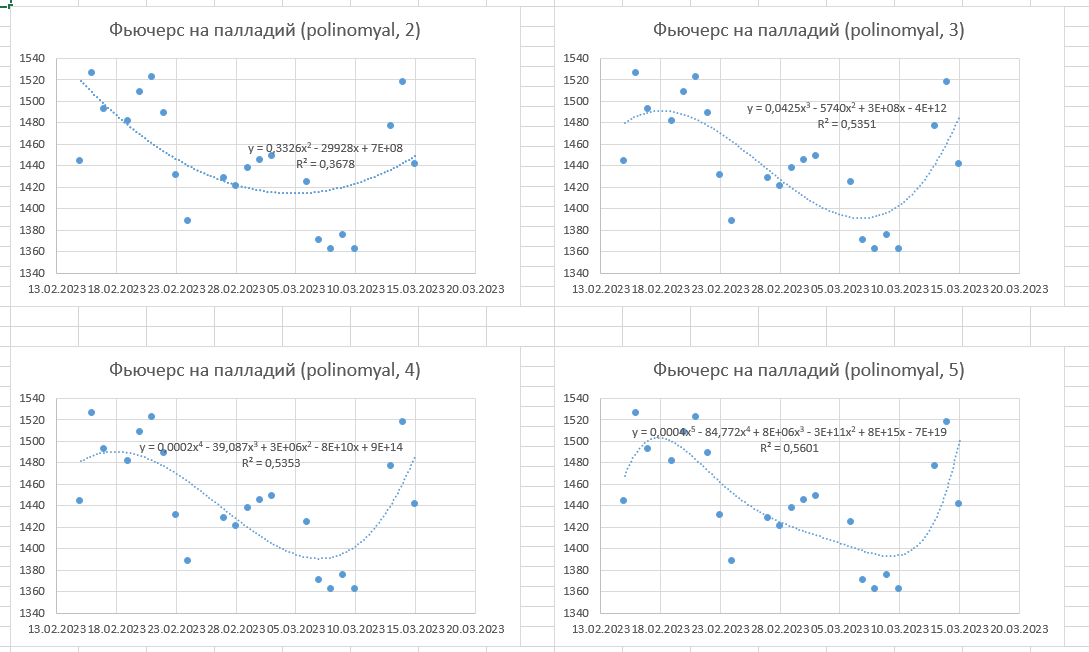
**Excel**

В Excel были построены линии тренда с использованием встроенных функций:

* Линейная, полиномиальная, степенная и логарифмическая аппроксимации.
* Для каждой модели отображены уравнение тренда и R^2.

Скриншоты графиков и уравнений:





**Python**

В Python для анализа использовались библиотеки numpy, scikit-learn, matplotlib и функции:

* Линейная регрессия (LinearRegression).
* Полиномиальная регрессия с использованием степени k = 6.
* Степенная модель с функцией y=a⋅x.
* Логарифмическая модель y=a⋅ln(x)+b.

**Код**

*def* plot\_second\_task():

*def* polynomial\_model(*x*, *degree*):

        poly = PolynomialFeatures(*degree*=*degree*)

        x\_poly = poly.fit\_transform(*x*.reshape(-1, 1))

        model = LinearRegression().fit(x\_poly, prices)

        return model.predict(x\_poly), model

*def* power\_law(*x*, *a*, *b*):

        return *a* \* *x* \*\* *b*

*def* log\_model(*x*, *a*, *b*):

        return *a* \* np.log(*x*) + *b*

    linear\_model = LinearRegression().fit(x.reshape(-1, 1), prices)

    linear\_pred = linear\_model.predict(x.reshape(-1, 1))

    degrees = [2, 3, 4, 5, 6]

    poly\_preds = []

    poly\_models = []

    r2\_polys = []

    for degree in degrees:

        pred, model = polynomial\_model(x, degree)

        poly\_preds.append(pred)

        poly\_models.append(model)

        r2\_polys.append(r2\_score(prices, pred))

    params\_power, \_ = curve\_fit(power\_law, x + 1, prices, *maxfev*=10000)

    power\_pred = power\_law(x + 1, \*params\_power)

    params\_log, \_ = curve\_fit(log\_model, x + 1, prices)

    log\_pred = log\_model(x + 1, \*params\_log)

    r2\_linear = r2\_score(prices, linear\_pred)

    r2\_power = r2\_score(prices, power\_pred)

    r2\_log = r2\_score(prices, log\_pred)

    fig, axs = plt.subplots(3, 2, *figsize*=(14, 15))

    axs[0, 0].scatter(x, prices, *color*='blue', *label*='Data')

    axs[0, 0].plot(x, linear\_pred, *color*='red', *label*=*f*'Linear fit: R^2={r2\_linear*:.4f*}')

    axs[0, 0].set\_title("Linear Fit")

    axs[0, 0].legend()

    for i, (degree, pred, r2) in enumerate(zip(degrees, poly\_preds, r2\_polys)):

        row = (i + 1) // 2

        col = (i + 1) % 2

        axs[row, col].scatter(x, prices, *color*='blue', *label*='Data')

        axs[row, col].plot(x, pred, *color*='green', *label*=*f*'Poly fit (degree={degree}): R^2={r2*:.4f*}')

        axs[row, col].set\_title(*f*"Polynomial Fit (degree {degree})")

        axs[row, col].legend()

    axs[0, 0].text(0.05, 0.9, *f*"y = {linear\_model.coef\_[0]*:.2f*}x + {linear\_model.intercept\_*:.2f*}",

*transform*=axs[0, 0].transAxes, *fontsize*=10, *bbox*=dict(*facecolor*='white', *alpha*=0.5))

*def* format\_polynomial(*coefficients*):

        terms = []

        for i, coef in enumerate(*coefficients*):

            if abs(coef) > 1e-6:

                if i == 0:

                    terms.append(*f*"{coef*:.2f*}")

                elif i == 1:

                    terms.append(*f*"{coef*:.2f*}x")

                else:

                    terms.append(*f*"{coef*:.2f*}x^{i}")

        return " + ".join(terms).replace("+ -", "- ")

    for i, (degree, model) in enumerate(zip(degrees, poly\_models)):

        row = (i + 1) // 2

        col = (i + 1) % 2

        poly\_coefficients = model.coef\_

        poly\_equation = format\_polynomial(poly\_coefficients)

        axs[row, col].text(0.05, 0.8, *f*"y = {poly\_equation}",

*transform*=axs[row, col].transAxes, *fontsize*=8, *bbox*=dict(*facecolor*='white', *alpha*=0.5))

    axs[2, 0].scatter(x, prices, *color*='blue', *label*='Data')

    axs[2, 0].plot(x, power\_pred, *color*='purple', *label*=*f*'Power law fit: R^2={r2\_power*:.4f*}')

    axs[2, 0].set\_title("Power Law Fit")

    axs[2, 0].legend()

    axs[2, 1].scatter(x, prices, *color*='blue', *label*='Data')

    axs[2, 1].plot(x, log\_pred, *color*='orange', *label*=*f*'Logarithmic fit: R^2={r2\_log*:.4f*}')

    axs[2, 1].set\_title("Logarithmic Fit")

    axs[2, 1].legend()

    print(*f*"Linear Model: R^2 = {r2\_linear*:.4f*}, Equation: y = {linear\_model.coef\_[0]*:.2f*}x + {linear\_model.intercept\_*:.2f*}")

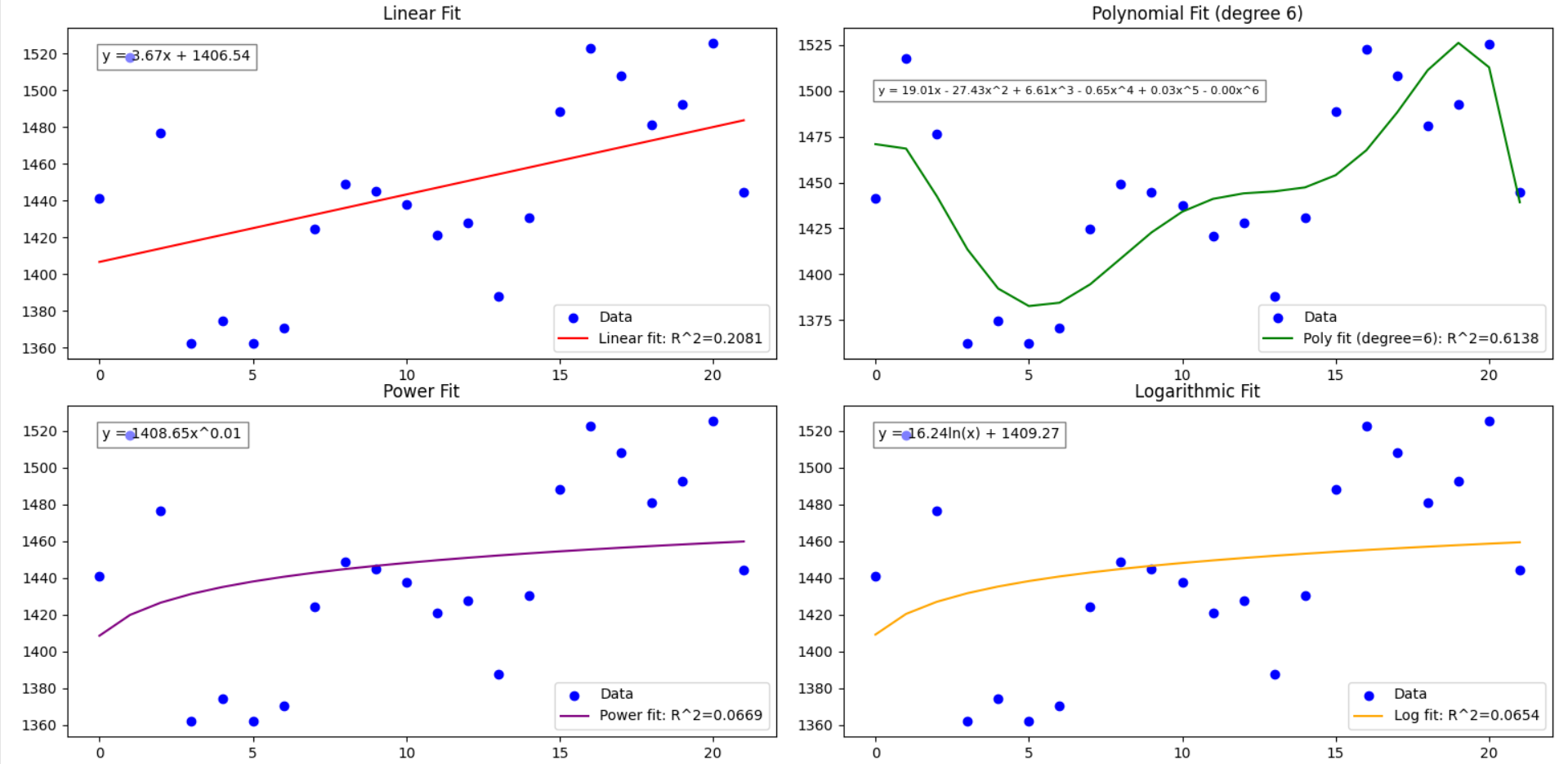
    print(*f*"Power Law Model: R^2 = {r2\_power*:.4f*}, Equation: y = {params\_power[0]*:.2f*}x^{params\_power[1]*:.2f*}")

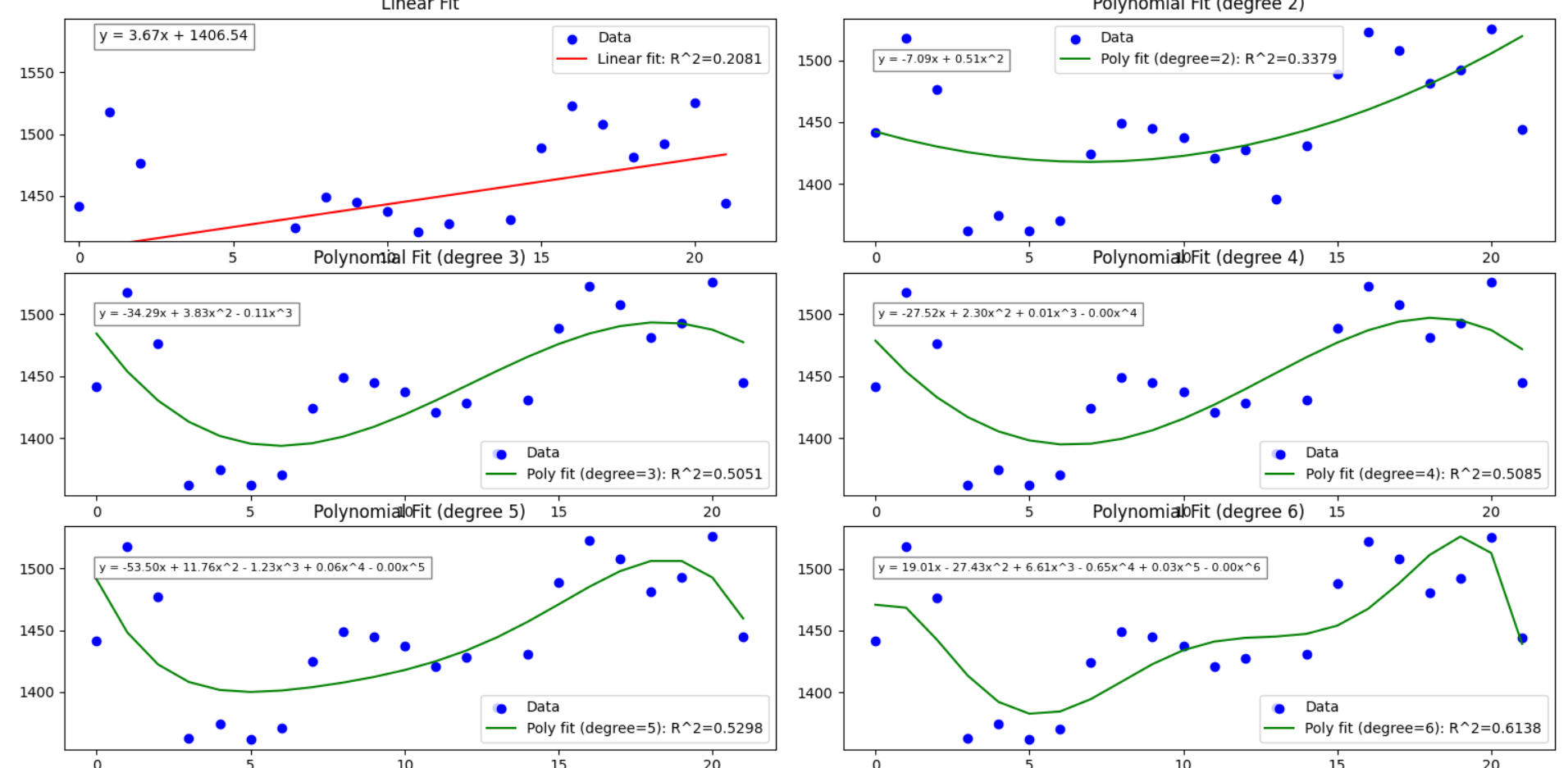
    print(*f*"Logarithmic Model: R^2 = {r2\_log*:.4f*}, Equation: y = {params\_log[0]*:.2f*}ln(x) + {params\_log[1]*:.2f*}")

    plt.tight\_layout()

    plt.show()

**Итоговые графики:**





## **Выводы**

**Различия между Excel и Python**  
Графики в Excel и Python могут отличаться из-за следующих факторов:

1. **Методы аппроксимации**: Excel использует собственные алгоритмы для расчета параметров тренда, которые могут отличаться от подходов в Python
2. **Числовая стабильность**: При расчетах коэффициентов уравнений и R^2 в Python могут использоваться более точные алгоритмы или настройки, влияющие на результаты.
3. **Преобразование данных**: В Python, чтобы избежать ошибок при работе со степенной и логарифмической аппроксимацией, к данным добавляется смещение (x+1). Это может влиять на параметры модели.

**Коэффициент детерминации (R^2)**

R^2 показывает, насколько хорошо модель объясняет вариацию данных. Значения R^2 для различных моделей из Python и Excel близки, но могут отличаться из-за разницы в расчетах.

**Вывод по моделям**

На основе коэффициента R^2, наиболее подходящей для данных является полиномиальная модель со степенью 6.

Линейная модель в Excel имеет наименьшее R^2, что говорит о её низкой точности для описания данных. В python же логарифмическая модель показала еще менее точный результат.

# **Общий вывод по лабораторной работе**

В ходе выполнения лабораторной работы были решены две задачи с использованием различных методов анализа и моделирования.

**Первая задача** касалась оптимизации производства на швейной фабрике. Были успешно построены и решены математические модели для максимизации прибыли.

Решение задачи выполнено двумя способами: с использованием MS Excel и библиотеки PuLP в Python.

Результаты обоих подходов совпали, что подтверждает корректность математической модели и её реализации.

Оптимальная стратегия производства определена, а максимальная стоимость продукции составила 2115 руб.

**Вторая задача** была направлена на анализ котировок и построение аналитических моделей с помощью Excel и Python.

Построены линии тренда с использованием различных типов аппроксимации (линейная, полиномиальная, степенная, логарифмическая).

В Python и Excel получены близкие, но не идентичные результаты из-за различий в методах расчёта и обработки данных.

Полиномиальная модель шестой степени оказалась наиболее подходящей для описания данных по коэффициенту детерминации R^2.

Различия в точности между платформами подчёркивают важность выбора инструмента в зависимости от задачи.

**Итог:**  
Работа показала, что применение математических моделей и современных инструментов анализа данных позволяет эффективно решать задачи оптимизации и анализа. Использование Python предоставляет более широкие возможности для автоматизации, обработки данных и построения сложных моделей.

**Сравнение линий тренда по коэффициенту детерминации (python) ()**

|  |  |  |
| --- | --- | --- |
| Тип линии тренда | Функция линии тренда | Коэффициент |
| Линейный | y=3,67x + 1406,54 | 0.2081 |
| Полиномиальный 2 степени | y = -7.09x + 0.51x^2 | 0.3379 |
| Полиномиальный 3 степени | y = -34.29x + 3.83x^2 - 0.11x^3 | 0.5051 |
| Полиномиальный 4 степени | y = -27.52x + 2.30x^2 + 0.01x^3 - 0.00x^4 | 0.5085 |
| Полиномиальный 5 степени | y = -53.50x + 11.76x^2 - 1.23x^3 + 0.06x^4 - 0.00x^5 | 0.5298 |
| Полиномиальный 6 степени | y = 19.01x - 27.43x^2 + 6.61x^3 - 0.65x^4 + 0.03x^5 - 0.00x^6 | 0.6138 |
| Экспоненциальный | y = 1408.65x^0.01 | 0.0669 |
| Логарифмический | y = 16.24ln(x) + 1409.27 | 0.0654 |

**Сравнение линий тренда по коэффициенту детерминации (excel) ()**

|  |  |  |
| --- | --- | --- |
| Тип линии тренда | Функция линии тренда | Коэффициент |
| Линейный | y = -2,5637x + 116777 | 0,1874 |
| Полиномиальный 2 степени | y = 0,3326x2 - 29928x + 7E+08 | 0,3678 |
| Полиномиальный 3 степени | y = 0,0425x3 - 5740x2 + 3E+08x - 4E+12 | 0,5351 |
| Полиномиальный 4 степени | y = 0,0002x4 - 39,087x3 + 3E+06x2 - 8E+10x + 9E+14 | 0,5353 |
| Полиномиальный 5 степени | y = 0,0004x5 - 84,772x4 + 8E+06x3 - 3E+11x2 + 8E+15x - 7E+19 | 0,5601 |
| Полиномиальный 6 степени | y = -0,0001x6 + 29,445x5 - 3E+06x4 + 2E+11x3 - 7E+15x2 + 1E+20x - 9E+23 | 0,7068 |
| Экспоненциальный | y = 1E+38e-0,002x | 0,1885 |
| Логарифмический | y = -1E+05ln(x) + 1E+06 | 0,1874 |

# **Ссылка на реализацию (github):**

https://github.com/ghimik/Computer-Simulating-Systems-Labs